LISTING OF CLAIMS:

Claims 1 – 30 (Canceled).

Claim 31 (Currently Amended): A method, in a mixed static and dynamic environment, for a virtual machine in which statically precompiled code may be securely executed by a virtual machine by means of a compiler or code generator, the method comprising the steps of:

a) saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment;

b) ~~a)~~ verifying that an intermediate or byte-code representation of the program is safe;

c) ~~b)~~ forming a secure hash describing the generated code;

d) ~~c)~~ forming a secure hash describing the byte code;

e) ~~d)~~ digitally signing the secure hashes of the generated code and the byte code;

the executing virtual machine reuses this code by

f) ~~e)~~ verifying that the secure hash of the byte-code matches the digitally signed secure hash for the byte-code;

g) ~~f)~~ verifying that the secure hash of the generated code matches the digitally signed secure hash for the generated code; and

h) ~~g)~~ loading and executing the generated code.

Claim 32 (Currently Amended): A method, in a mixed static and dynamic environment, for linking separately statically, precompiled code at run-time within a virtual machine by modifying the code, the method comprising the steps of

using a compiler to perform the steps of

a) saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment;

b) ~~a)~~ maintaining symbolic entries for externally referenced symbols; and

c) ~~b)~~ maintaining a mapping from locations in the generated code that reference external symbols to the symbolic entry for that symbol;

and the virtual machine, before the code is executed, performs the steps of

d) ~~c)~~ using the mapping and symbolic entries created by the compiler to generate direct references in the generated code to the externally referenced symbols that have been resolved by the virtual machine; and

e) ~~d)~~ performing the default action on those external symbols that have not been resolved.

Claim 33 (Currently Amended): A method, in a mixed static and dynamic environment, for updating statically generated precompiled code (C), at run-time, when separately compiled code (S), which contained symbols referenced by C changes, the method comprising the steps of:

saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment;

having a compiler generating the code for S to a) associate with method and data names, or signatures, in S a secure hash of the name of the method and data names or signatures in S with the compiler for C recording the secure hash for the byte code corresponding to any S that affects the code generated for C; and

having the virtual machine executing C to

b) check if any byte codes associated with any names in the S relied upon by C have changed by comparing the secure hash of the names associated with S with the secure hash stored for this byte code in C, and

c) dynamically recompile the byte codes associated with C if any byte codes associated with S have changed.

Claim 34 (Currently Amended): A method, in a mixed static and dynamic environment, for maintaining full compliance with a language requiring dynamic compilation without requiring the overhead of a just-in-time compiler to be present in the virtual machine, while enabling the use of statically generated precompiled code (C) for some byte code that depends on some byte code (S) that may be separately compiled,

saving pre-compiled programs, including determining where to place said programs, annotating the programs with dependent information, annotating the programs with

dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment;

having a compiler generating code for S

a) associate with S a secure hash of the byte code associated with S;

having the compiler for C to

b) record the secure hash for the byte code corresponding to any S that affects the code generated for C; and

having the virtual machine executing C to

c) check if any byte codes associated with any code S relied upon by C have changed by comparing the secure hash of the byte code associated with S with the secure hash stored for this byte code with C; and

d) interpret the byte codes corresponding to C.

Claim 35 (Previously Presented: A method according to claim 31, comprising the further step of the compiler performing dependence checks during program execution to avoid using a stale code for a procedure in the event of changes to other codes.

Claim 36 (Previously Presented): A method according to claim 35, wherein the step of performing dependence checks includes using time stamps to determine if any of the classes on which the code is dependent have changed.